Theoretical Computer Science 119 (1993) 103–125
Elsevier

On a generalized product for domains

Walter Dosch

Institut für Mathematik, Universität Augsburg, D-86135 Augsburg, Germany

Abstract

Dosch, W., On a generalized product for domains, Theoretical Computer Science 119 (1993) 103–125.

We introduce a new product operation for domains, called *I-product*, uniformly covering the cartesian and the smash products. We discuss its order-theoretic properties and give different axiomatizations for it. We also investigate the set of continuous functions from and into the *I*-product. Finally, we study the lattice of *I*-strict continuous extensions of operations on sets to the corresponding flat domains. In summary, the *I*-product preserves most of the well-known properties of the domain products while offering greater flexibility by incorporating strictness constraints into domain constructions.

1. Introduction

In the denotational description of programming languages, the meaning of a program is modeled by an abstract mathematical entity in a semantic domain. To render the meaning function total, the "value" of a nonterminating computation is denoted by a special element, called $\perp$. The meaning function is defined compositionally along the syntactic structure of the program, i.e., the denotation of a composite phrase is a function of the denotations of its subphrases. More formally, the meaning function is the unique homomorphism from the syntactic into the semantic algebra; by initiality of the syntactic algebra, it is unique (cf. [6]). Therefore, the essential semantic design decisions for a programming language are not determined by the interpretation function itself, but by the choice of a specific semantic algebra.

For interpreting the different syntactic categories of a programming language, various composite domains are needed. Usually, these domains are not defined anew by their own; rather they are constructed from simpler domains using some basic domain constructions, for example, forming the direct sum, the direct product, the

Correspondence to: W. Dosch, Institut für Mathematik, Universität Augsburg, D-86135 Augsburg, Germany. Email: dosch@uni-augsburg.dc.

function space, or the power domain. In this way, domain constructions combine semantic algebras as the basic building blocks of a modular semantic description (cf. [14]). Here the semantic design decisions, being specific for a programming language, are reflected by using particular domain constructions. Thereby the envisaged semantic properties are "frozen" into the domain itself.

For the product of domains, there are the cartesian product forming the set of all tuples of its components, and the smash product where all tuples containing $\perp$ are identified. The smash product is adequate for describing the argument domain of routines that are strict in all of their arguments, whereas the cartesian product is necessary for routines that are strict in none of their arguments. However, these are just two borderline cases of a more general situation: Multiary routines can be strict in some and nonstrict in other arguments.

To this end, we introduce the I-product as a new domain construction. This is a generalized product operation for domains that covers uniformly both the cartesian and the smash products. It adequately models products of domains where the tuple constructor is strict in an arbitrary subset I of its arguments. The I-product preserves most of the properties of the cartesian resp. of the smash product while offering a greater flexibility in expressing particular strictness constraints.

The paper is organized as follows: In Section 2 we introduce the I-product together with its canonical operations by a concrete domain model. Then we show that the I-product inherits all major properties from the component domains. In Section 3 we investigate the structure of the set of all I-products belonging to different index sets: under mild restrictions it forms a complete lattice where the smash product is the zero element, and the cartesian product is the unit element. In Section 4 we give an internal axiomatization where the I-product is characterized uniquely up to isomorphism by the algebraic properties of its operations. In the external axiomatization to follow, the I-product is defined by functions from external trial domains. In Section 5 we investigate the properties of functions from and into the I-product; in particular, we relate strict unary functions on the I-product with I-strict multiary functions on the cartesian product. In Section 6 we discuss how multiary operations on sets can be extended to I-strict continuous functions on the corresponding flat partial orders. The set of all I-strict extensions of an operation forms a complete lattice. This leads to new results about the existence of greatest I-strict continuous extensions. In Section 7 we finally outline the application of the I-product to the denotational description of programming languages and to the solution of recursive domain equations.

We assume that the reader is familiar with the denotational description of programming languages and the underlying domain theory. Nevertheless, to render the paper self-contained, in the appendix we briefly survey all basic notions and results used. For more details on the order-theoretic foundations of denotational semantics, the interested reader is referred, for example, to [11] or [26]. For the foundations of domain theory one should examine [16, 20, 21]. Different classes of domains are investigated in [8, 22–24]. Introductions to lattice theory are given in [3, 4, 10]. Excellent up-to-date surveys on domain theory resp. on denotational semantics are provided by [9, 15].

2. Definition of the I-product

In this section we introduce the I-product together with its canonical operations by giving a concrete domain model.

Definition 2.1. Let $(M_j, \sqsubseteq_j)\,(j\in[1,n])$ be $n\geqslant 0$ posets with least elements $\perp_j$. For an index set $I\subseteq[1,n]$ the *I-product* $M_1 \overset{I}{\star}\cdots\star M_n$ is the set

$$\{\langle x_1,\ldots,x_n\rangle \mid \bigwedge_{i\in I} x_i\in M_i\setminus\{\perp_i\} \wedge \bigwedge_{j\in[1,n]\setminus I} x_j\in M_j\}\cup\{\langle \perp_1,\ldots,\perp_n\rangle\}$$

equipped with the order relation

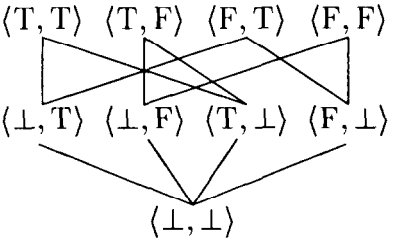
$$\langle x_1,\ldots,x_n\rangle \sqsubseteq \langle y_1,\ldots,y_n\rangle \text{ iff } x_j\sqsubseteq_j y_j \text{ for all } j\in[1,n].$$

For $I=\emptyset$ we obtain the *cartesian product* $M_1\times\cdots\times M_n$ and for $I=[1,n]$ the *smash product* $M_1\otimes\cdots\otimes M_n$.

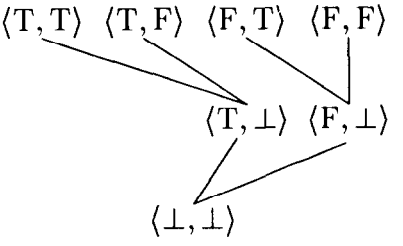
The I-product comprises all those tuples for which, for all $i\in I$, the ith component is different from $\perp_i$. The tuple $\langle\perp_1,\ldots,\perp_n\rangle$ represents the equivalence class of all those les having $\perp_i$ as ith component for some $i\in I$. For $n=0$ the $\emptyset$-product is the singleton set $\{\langle\rangle\}$. In the sequel, we frequently use the vector notation $\vec{x}=\langle x_1,\ldots,x_n\rangle$ to denote tuples.

Examples 2.2. Let $\mathscr{B}=\{\mathrm{T},\mathrm{F}\}$ be the set of Boolean values, and $\mathscr{B}^\perp=\{\perp,\mathrm{T},\mathrm{F}\}$ the associated flat poset.

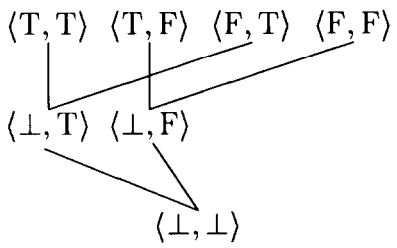
(a) Cartesian product $\mathscr{B}^\perp\times\mathscr{B}^\perp=\mathscr{B}^\perp\overset{\emptyset}{\star}\mathscr{B}^\perp$



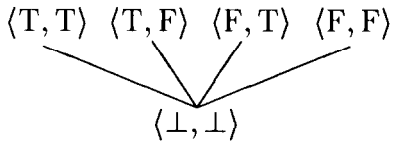
(b) Left-strict product $\mathscr{B}^\perp\overset{\{1\}}{\star}\mathscr{B}^\perp$



(c) Right-strict product $\mathscr{B}^{\perp} \overset{\{2\}}{\star} \mathscr{B}^{\perp}$



(d) Smash product $\mathscr{B}^{\perp} \otimes \mathscr{B}^{\perp} = \mathscr{B}^{\perp} \overset{\{1,2\}}{\star} \mathscr{B}^{\perp}$



Since all its components M_j are posets with least elements $\perp_j$, the I-product $M_1 \star \overset{I}{\ldots} \star M_n$ forms again a poset under the componentwise order with least element $\vec{\perp}$. On the I-product we now define the canonical operations.

Definition 2.3. Let $(M_j, \sqsubseteq_j)\,(j \in [1,n])$ be posets with least elements $\perp_j$ and $I \subseteq [1,n]$ an index set. For every $j \in [1,n]$ the *jth projection* $\pi_j : M_1 \star \overset{I}{\ldots} \star M_n \to M_j$ is given by $\pi_j(\vec{x}) = x_j$. The *construction* $cons^I : M_1 \times \cdots \times M_n \to M_1 \star \overset{I}{\ldots} \star M_n$ is defined by

$$cons^I(x_1, \ldots, x_n) = \begin{cases} \vec{x} & \text{if } x_i \neq \perp_i \text{ for all } i \in I, \\ \vec{\perp} & \text{if } x_i = \perp_i \text{ for some } i \in I. \end{cases}$$

The projections and the construction are monotone mappings. In the I-product the least upper bounds of directed sets are determined componentwise.

Proposition 2.4. *Let $(M_j, \sqsubseteq_j)$ $(j \in [1,n])$ be posets, $I \subseteq [1,n]$ an index set, and $S \subseteq M_1 \star \overset{I}{\ldots} \star M_n$. Then $\bigsqcup S$ exists iff $\bigsqcup_j \pi_j(S)$ exists for all $j \in [1,n]$. Moreover, if $\bigsqcup S$ exists, then $\bigsqcup S = cons^I(\bigsqcup_1 \pi_1(S), \ldots, \bigsqcup_n \pi_n(S))$.*

Proof. Assume that $\bigsqcup S = \vec{s}$ exists. We show $\bigsqcup_j \pi_j(S) = s_j$ for all $j \in [1,n]$. *Upper bound*: For $x_j \in \pi_j(S)$ there exists $\vec{x} \in S$. From $\vec{x} \sqsubseteq \vec{s}$ we get $x_j \sqsubseteq_j s_j$. This shows $\pi_j(S) \sqsubseteq_j s_j$. *Least upper bound*: Assume that $\pi_j(S) \sqsubseteq_j y_j \in M_j$ for all $j \in [1,n]$. If $y_i = \perp_i$ for some $i \in I$, then $S = \{\vec{\perp}\}$, and $s_j = \perp_j \sqsubseteq_j y_j$. Otherwise, $S \sqsubseteq \vec{y} \in M_1 \star \overset{I}{\ldots} \star M_n$; hence, $\vec{s} \sqsubseteq \vec{y}$ holds and thus $s_j \sqsubseteq_j y_j$.

Conversely, assume that $s_j = \bigsqcup_j \pi_j(S)$ exists for all $j \in [1,n]$. Put $S_j = \pi_j(S)$. If $s_i = \perp_i$ for some $i \in I$, then $S = \{\vec{\perp}\}$, and $\bigsqcup S = \vec{\perp}$ exists. Otherwise, $\vec{s} \in M_1 \star \overset{I}{\ldots} \star M_n$, and we

show $\bigsqcup S = \vec{s}$. *Upper bound*: Let $\vec{x} \in S$. Then $x_j \in S_j$, thus $x_j \sqsubseteq_j s_j$ and hence $\vec{x} \sqsubseteq \vec{s}$. This shows $S \sqsubseteq \vec{s}$. *Least upper bound*: Assume $S \sqsubset \vec{y}$. Then $S_j \sqsubseteq_j y_j$ holds, therefore, $s_j \sqsubseteq_j y_j$ ($j \in [1, n]$) and hence $\vec{s} \sqsubseteq \vec{y}$.

This also establishes the equation $\bigsqcup S = cons^I(\bigsqcup_1 \pi_1(S), \ldots, \bigsqcup_n \pi_n(S))$. □

Now we have collected all prerequisites to show that the I-product of cpos again forms a cpo.

Proposition 2.5. *Let $(M_j, \sqsubseteq_j, \bot_j)$ ($j \in [1, n]$) be complete partial orders and $I \subseteq [1, n]$. Then $(M_1 \star \overset{I}{\ldots} \star M_n, \sqsubseteq, \vec{\bot})$ forms a complete partial order. The projections and the construction are continuous.*

Proof. Let $D \subseteq M_1 \star \overset{I}{\ldots} \star M_n$ be directed. Then $\pi_j(D) \subseteq M_j$ is directed; hence $\bigsqcup_j \pi_j(D)$ exists for all $j \in [1, n]$, and by Proposition 2.4 $\bigsqcup D$ exists as well. The continuity of π_j and $cons^I$ follows from Proposition 2.4. □

If all components are posets where every directed set is finite, then the I-product also has only finite directed sets, and thus forms a complete partial order.

The finite elements of the I-product can be characterized simply: the basis of the I-product is the product of the bases of its components.

Proposition 2.6. $B(M_1 \star \overset{I}{\ldots} \star M_n) = B(M_1) \star \overset{I}{\ldots} \star B(M_n)$.

Proof. Assume $\vec{x} \in B(M_1 \star \overset{I}{\ldots} \star M_n)$. For all $j \in [1, n]$ let $D_j \subseteq M_j$ be directed with $x_j \sqsubseteq_j \bigsqcup_j D_j$. Then $D = D_1 \star \overset{I}{\ldots} \star D_n$ is directed in $M_1 \star \overset{I}{\ldots} \star M_n$ with $\vec{x} \sqsubseteq \bigsqcup D$. Since $\vec{x}$ is finite, we have $\vec{x} \sqsubseteq \vec{d}$ for some $\vec{d} \in D$; thus, $x_j \sqsubseteq_j d_j$. This shows $x_j \in B(M_j)$ for every $j \in [1, n]$; therefore, $\vec{x} \in B(M_1) \star \overset{I}{\ldots} \star B(M_n)$.

Conversely, assume $\vec{x} \in B(M_1) \star \overset{I}{\ldots} \star B(M_n)$. Let $D \subseteq M_1 \star \overset{I}{\ldots} \star M_n$ be directed with $\vec{x} \sqsubseteq \bigsqcup D$. Then $x_j \sqsubseteq_j \bigsqcup_j \pi_j(D)$ holds. Since $x_j \in M_j$ is finite, there is $\vec{d}^j \in D$ with $x_j \sqsubseteq_j \pi_j(\vec{d}^j)$ ($j \in [1, n]$). Since D is directed, there is $\vec{e} \in D$ with $\{\vec{d}^1, \ldots, \vec{d}^n\} \sqsubseteq \vec{e}$. Then $\vec{x} \sqsubseteq \vec{e}$ holds, which shows $\vec{x} \in B(M_1 \star \overset{I}{\ldots} \star M_n)$. □

Besides the finiteness, the I-product also inherits the consistent completeness and the countable algebraicity from its components. In summary, we therefore obtain the following theorem.

Theorem 2.7. *If all components are domains, then their I-product forms a domain as well.*

Proof. *Consistent completeness*: Let $S \subseteq M_1 \star \overset{I}{\ldots} \star M_n$ be consistent: $S \sqsubseteq \vec{x}$. Then $\pi_j(S) \sqsubseteq_j x_j$ holds; hence, by assumption, $\bigsqcup_j \pi_j(S)$ exists for every $j \in [1, n]$, and, by Proposition 2.4, $\bigsqcup S$ exists as well.

Countable basis: By assumption every $B(M_j)$ is countable; hence, by Proposition 2.6, $B(M_1 \star \overset{I}{\ldots} \star M_n)$ is countable as well.

Algebraicity: From Proposition 2.6 we get $B(M_1 \star \overset{I}{\ldots} \star M_n, \vec{y}) = B(M_1, y_1) \star \overset{I}{\ldots} \star B(M_n, y_n)$. Thus, we have $\bigsqcup B(M_1 \star \overset{I}{\ldots} \star M_n, \vec{y}) = cons^I(\bigsqcup_1 B(M_1, y_1), \ldots, \bigsqcup_n B(M_n, y_n)) = \vec{y}$. □

3. Relations between different I-products

In this section we investigate the relationship between the I-products belonging to different index sets. The larger the index set grows, more the tuples are identified with the least element. This leads to a lattice structure on the set of all I-products.

Definition 3.1. Let $(M_j, \sqsubseteq_j)$ $(j \in [1,n])$ be posets with least elements $\perp_j$ and $I \subseteq J \subseteq [1,n]$ index sets. The *inclusion* $i^{JI}: M_1 \star \overset{J}{\ldots} \star M_n \to M_1 \star \overset{I}{\ldots} \star M_n$ is given by $i^{JI}(\vec{x}) = \vec{x}$. The *construction* $cons^{IJ}: M_1 \star \overset{I}{\ldots} \star M_n \to M_1 \star \overset{J}{\ldots} \star M_n$ is defined as

$$cons^{IJ}(\vec{x}) = \begin{cases} \vec{x} & \text{if } x_i \neq \perp_i \text{ for all } i \in J, \\ \vec{\perp} & \text{if } x_i = \perp_i \text{ for some } i \in J. \end{cases}$$

Again the inclusions and the constructions are monotone and continuous mappings. Corresponding inclusion and construction mappings form an embedding pair, i.e.

$$i^{JI} \circ cons^{IJ} \sqsubseteq id_{M_1 \star \overset{I}{\ldots} \star M_n} \quad \text{and} \quad cons^{IJ} \circ i^{JI} = id_{M_1 \star \overset{J}{\ldots} \star M_n}.$$

The set of all inclusion mappings as well as the set of all construction mappings are closed under function composition, since we have for all index sets $I \subseteq J \subseteq K \subseteq [1,n]$

$$i^{JI} \circ i^{KJ} = i^{KI}.$$

$$cons^{JK} \circ cons^{IJ} = cons^{IK}.$$

The set of all I-products is closed under intersection, but not under union.

Corollary 3.2. *For all $I, J \subseteq [1,n]$, we have*

$$M_1 \star \overset{I}{\ldots} \star M_n \cap M_1 \star \overset{J}{\ldots} \star M_n = M_1 \star \overset{I \cup J}{\ldots} \star M_n,$$

$$M_1 \star \overset{I}{\ldots} \star M_n \cup M_1 \star \overset{J}{\ldots} \star M_n \subseteq M_1 \star \overset{I \cap J}{\ldots} \star M_n.$$

Proof. Straightforward. □

Given a fixed set of nonsingleton components, the set of all I-products bears a lattice structure.

Theorem 3.3. *Let* $(M_j, \sqsubseteq_j)$ $(j \in [1,n])$ *be nonsingleton posets with least elements* $\perp_j$. *Define on the set* $\mathscr{M} = \{M_1 \star \overset{I}{\ldots} \star M_n \mid I \subseteq [1,n]\}$ *of all I-products the operations*

$$M_1 \star \overset{I}{\ldots} \star M_n \sqcup M_1 \star \overset{J}{\ldots} \star M_n = M_1 \star \overset{I \cap J}{\ldots} \star M_n,$$

$$M_1 \star \overset{I}{\ldots} \star M_n \sqcap M_1 \star \overset{J}{\ldots} \star M_n = M_1 \star \overset{I \cup J}{\ldots} \star M_n,$$

$$\overline{M_1 \star \overset{I}{\ldots} \star M_n} = M_1 \star \overset{[1,n]\setminus I}{\ldots} \star M_n.$$

Then $(\mathscr{M}, \sqcup, \sqcap, \bar{\ })$ *forms a Boolean lattice where the cartesian product is the unit element, and the smash product is the zero element.*

Proof. Define $\sigma : \mathscr{P}([1,n]) \to \mathscr{M}$ by $\sigma(I) = M_1 \star \overset{I}{\ldots} \star M_n$. Since all M_j are nonsingleton posets, σ is injective; hence, the meet, join, and complement operation on $\mathscr{M}$ are well defined. σ defines a lattice isomorphism between the powerset $(\mathscr{P}([1,n]), \cap, \cup, \mathscr{C})$ and $(\mathscr{M}, \sqcup, \sqcap, \bar{\ })$. □

The subcpo property of I-products can be characterized in different ways.

Proposition 3.4. *Let* $(M_j, \sqsubseteq_j, \perp_j)$ $(j \in [1,n])$ *be nonsingleton complete partial orders and* $I, J \subseteq [1,n]$. For two products $P = M_1 \star \overset{I}{\ldots} \star M_n$ *and* $Q = M_1 \star \overset{J}{\ldots} \star M_n$ *in* $\mathscr{M}$ *the following statements are equivalent*:

(1) P *is a subcpo of* Q.

(2) $I \supseteq J$.

(3) $P \geqslant Q$ *holds in the lattice order.*

Proof. (1) ⇒ (2): Assume that P is a subcpo of Q. Then $P \subseteq Q$ holds, hence $I \supseteq J$.

(2) ⇒ (3): Assume $I \supseteq J$. Then $P \sqcap Q = P$ holds, hence $P \geqslant Q$.

(3) ⇒ (1): Assume $P \geqslant Q$. Then $P \sqcap Q = P$ holds, thus $I \supseteq J$ and therefore $P \subseteq Q$. Let $D \subseteq P$ be directed and $s_j = \bigsqcup_j \pi_j(D)$ $(j \in [1,n])$. By Proposition 2.4, $\bigsqcup_Q D = cons^J(s_1, \ldots, s_n)$. If $s_i = \perp_i$ for some $i \in I$, then $\bigsqcup_Q D = \vec{\perp} \in P$. Otherwise $\bigsqcup_Q D = \vec{s} \in P$. □

In particular, for fixed components the smash product is a subcpo of every I-product, which itself is a subcpo of the cartesian product.

If some cpos are singleton sets, then different index sets may lead to the same product domain.

Example 3.5. Let $M_1 = \{\perp_1\}$, $M_2 = \{\perp_2, 0\}$, and $M_3 = \{\perp_3, \mathrm{T}, \mathrm{F}\}$. The lattice $\mathscr{M}$ has five elements, since $M_1 \star \overset{\{1\}}{\ldots} \star M_3 = M_1 \star \overset{\{1,2\}}{\ldots} \star M_3 = M_1 \star \overset{\{1,3\}}{\ldots} \star M_3 = M_1 \star \overset{\{1,2,3\}}{\ldots} \star M_3$ are all singleton sets. $\mathscr{M}$ forms a noncomplemented distributive lattice.

Finally, note that every I-product can solely be expressed in terms of the cartesian product and of the smash product using a dyadic left- (or right-) strict product operation.

4. Axiomatization of the I-product

The soundness of programming calculi widely depends on the properties of the domains underlying the programming language. In concrete domain models, as given in Section 2, such properties are implicitly contained in the representation of the semantic elements, whereas axiomatic definitions state them explicitly. These characteristic properties are determined by the algebraic laws of the operations for manipulating the semantic elements. For an axiomatic definition of domains within the framework of category theory see [5], and for their specification by continuous abstract types, cf. [13].

4.1. Internal axiomatization

First we collect some characteristic properties of the I-product which serve as a guideline for the internal axiomatization.

Proposition 4.1. *Let* $(M_j, \sqsubseteq_j)\ (j\in[1,n])$ *be posets with least elements* $\perp_j$ *and* $I\subseteq[1,n]$ *an index set.*

(a) *For all* $x_1\in M_1,\dots,x_n\in M_n$ *we have*

$$\pi_j(cons^I(x_1,\dots,x_n))=\begin{cases}x_j & \text{if } x_i\neq\perp_i \text{ for all } i\in I,\\ \perp_j & \text{if } x_i=\perp_i \text{ for some } i\in I\end{cases}\quad (j\in[1,n]).$$

(b) *For all* $\vec{x}\in M_1\overset{I}{\star}\dots\star M_n$ *we have*

$$cons^I(\pi_1(\vec{x}),\dots,\pi_n(\vec{x}))=\vec{x}.$$

Proof. Straightforward. □

The internal axiomatization follows the specification methodology of abstract data types: the I-product is defined in terms of its components. The following unifies the well-known axiomatization of the cartesian product and of the smash product; cf., for example, [16].

Theorem 4.2. *Let* $(M_j, \sqsubseteq_j, \perp_j)\ (j\in[1,n])$ *be* $n\geqslant 1$ *complete partial orders and* $I\subseteq[1,n]$. *Then there exists – up to isomorphism – a unique complete partial order* P *together with continuous functions*

$$c: M_1\times\cdots\times M_n\to P,$$

$$p_j: P\to M_j\quad (j\in[1,n])$$

such that for all $x_1\in M_1,\dots,x_n\in M_n$, *and* $x\in P$ *the following holds*:

(1) $$p_j(c(x_1,\dots,x_n))=\begin{cases}x_j & \text{if } x_i\neq\perp_j \text{ for all } i\in I,\\ \perp_i & \text{if } x_i=\perp_i \text{ for some } i\in I\end{cases}\quad (j\in[1,n]),$$

(2) $$c(p_1(x),\dots,p_n(x))=x.$$

Proof. *Existence*: Set $P = M_1 \star \overset{I}{\ldots} \star M_n$, $c = cons^I$, $p_j = \pi_j$ $(j \in [1,n])$, and apply Propositions 2.5 and 4.1.

Uniqueness: Define $f: P \to M_1 \star \overset{I}{\ldots} \star M_n$ by $f(x) = cons^I(p_1(x), \ldots, p_n(x))$ and $g: M_1 \star \overset{I}{\ldots} \star M_n \to P$ by $g(\vec{y}) = c(\pi_1(\vec{y}), \ldots, \pi_n(\vec{y}))$. Then f and g are continuous. We have $f(g(\vec{y})) = f(c(y_1, \ldots, y_n)) \overset{(1)}{=} cons^I(y_1, \ldots, y_n) = \vec{y}$ for all $\vec{y} \in M_1 \star \overset{I}{\ldots} \star M_n$, and $g(f(x)) = g(cons^I(p_1(x), \ldots, p_n(x))) = c(p_1(x), \ldots, p_n(x)) \overset{(2)}{=} x$ for all $x \in P$. Thus, $f \circ g = id_{M_1 \star \overset{I}{\ldots} \star M_n}$ and $g \circ f = id_P$ holds; hence P and $M_1 \star \overset{I}{\ldots} \star M_n$ are isomorphic. □

4.2. External axiomatization

In an external axiomatization the I-product is defined by the properties of functions from external trial domains. Functions leading into the components can only be combined to a single function leading into the I-product if they are compatible in the following sense.

Definition 4.3. Let $I \subseteq [1,n]$. A family $(f_j: M \to M_j)_{j \in [1,n]}$ of functions from a poset M with least element $\perp$ into posets M_j with least elements $\perp_j$ is called *I-consistent*, if, for all $x \in M$,

$$\bigvee_{i \in I} f_i(x) = \perp_i \quad \text{implies} \quad \bigwedge_{j \in [1,n]} f_j(x) = \perp_j.$$

As a guideline for the external axiomatization, we first state the following universal property.

Proposition 4.4. *Let $(M_j, \sqsubseteq_j, \perp_j)$ $(j \in [1,n])$ be complete partial orders and $I \subseteq [1,n]$. For all complete partial orders M and all I-consistent functions $(f_j: M \to M_j)_{j \in [1,n]}$, there exists a unique function $f: M \to M_1 \star \overset{I}{\ldots} \star M_n$ with $f_i = \pi_j \circ f$ $(j \in [1,n])$.*

Proof. *Existence*: Put $f(x) = cons^I(f_1(x), \ldots, f_n(x))$. If $f_i(x) = \perp_i$ for some $i \in I$, then $f(x) = \vec{\perp}$; hence $\pi_j(f(x)) = \perp_j = f_j(x)$ by the I-consistency $(j \in [1,n])$. Otherwise, $f_i(x) \neq \perp_i$ for all $i \in I$, and $\pi_j(f(x)) = f_j(x)$ holds for all $j \in [1,n]$.

Uniqueness: Let $g, h: M \to M_1 \star \overset{I}{\ldots} \star M_n$ be as assumed. Then $\pi_j \circ g = f_j = \pi_j \circ h$ holds for all $j \in [1,n]$; hence $g = h$. □

We now introduce the corresponding tupling operation on functions. In functional languages like FP [1], the function construction belongs to the basic program forming operations.

Definition 4.5. Let $(M_j, \sqsubseteq_j)$ $(j \in [0,n])$ be partial orders with least elements $\perp_j$ and $I \subseteq [1,n]$. Then the *function construction*

$$constr^I: [M_0 \to M_1] \times \cdots \times [M_0 \to M_n] \to [M_0 \to M_1 \star \overset{I}{\ldots} \star M_n]$$

is defined by $constr^I(f_1, \ldots, f_n)(x) = cons^I(f_1(x), \ldots, f_n(x))$ $(x \in M_0)$.

The function construction preserves the semantic properties of its argument functions; it also has a simple recombination property.

Corollary 4.6. *If all functions* $(f_j: M_0 \to M_j)_{j\in[1,n]}$ *are strict (monotone, continuous), then* $constr^I(f_1,\ldots,f_n)$ *is again strict (monotone, continuous). Moreover, for all functions* $f: M_0 \to M_1 \star \overset{I}{\ldots} \star M_n$ *we have* $f = constr^I(\pi_1 \circ f, \ldots, \pi_n \circ f)$.

Proof. The composition of strict (monotone, continuous) functions is strict (monotone, continuous). Moreover, for all $x\in M_0$ we have $constr^I(\pi_1 \circ f,\ldots,\pi_n \circ f)(x) = cons^I(\pi_1(f(x)),\ldots,\pi_n(f(x))) = f(x)$ by Proposition 4.1(b). □

The universal property above leads to the following external axiomatization where the I-product is characterized by the properties of functions from an external trial domain.

Theorem 4.7. *Let* $(M_j, \sqsubseteq_j, \perp_j)$ $(j\in[1,n])$ *be complete partial orders and* $I\subseteq[1,n]$. *Then there exists – up to isomorphism – a unique complete partial order P together with I-consistent continuous functions* $(p_j: P\to M_j)_{j\in[1,n]}$ *such that for all complete partial orders M and all I-consistent (strict) continuous functions* $(f_j: M\to M_j)_{j\in[1,n]}$ *there is a unique (strict) continuous function* $f: M\to P$ *with* $f_j = p_j \circ f$ $(j\in[1,n])$.

Proof. *Existence*: Set $P = M_1 \star \overset{I}{\ldots} \star M_n$, $p_j = \pi_j$ $(j\in[1,n])$, $f = constr^I(f_1,\ldots,f_n)$, and apply Proposition 4.4 and Corollary 4.6.

Uniqueness: Let $(P_1;\ p_1^1,\ldots,p_n^1)$ and $(P_2;\ p_1^2,\ldots,p_n^2)$ be as assumed. Setting $M = P_1$, $P = P_2$, and $f_j = p_j^1$ $(j\in[1,n])$, we obtain $f: P_1 \to P_2$ with $p_j^1 = p_j^2 \circ f$ $(j\in[1,n])$. Setting $M = P_2$, $P = P_1$, and $f_j = p_j^2$ $(j\in[1,n])$, we obtain $g: P_2 \to P_1$ with $p_j^2 = p_j^1 \circ g$ $(j\in[1,n])$. This shows $p_j^1 = p_j^1 \circ g \circ f$. Moreover, setting $= P_1$, $P = P_1$, and $f_j = p_j^1$, we obtain $p_j^1 = p_j^1 \circ id_{P_1}$; hence, by uniqueness, $g \circ f = id_{P_1}$. With a similar argument we obtain $f \circ g = id_{P_2}$. Thus, P_1 and P_2 are isomorphic. □

Replacing the uniqueness part in the above theorem by the recombination property of the function construction, we obtain the following axiomatization.

Corollary 4.8. *Let* $(M_j, \sqsubseteq_j, \perp_j)$ $(j\in[1,n])$ *be complete partial orders and* $I\subseteq[1,n]$. *Then there exists (up to isomorphism) a unique complete partial order P together with I-consistent continuous functions* $(p_j: P\to M_j)_{j\in[1,n]}$ *such that for all complete partial orders M the following holds:*

(1) *For all I-consistent (strict) continuous functions* $(f_j: M\to M_j)_{j\in[1,n]}$ *there is a (strict) continuous function* $f: M\to P$ *with* $f_j = p_j \circ f$ $(j\in[1,n])$. *Denote f by* $c^I(f_1,\ldots,f_n)$.

(2) *For all continuous functions* $h: M\to P$ *we have* $h = c^I(p_1 \circ h,\ldots,p_n \circ h)$.

5. Functions from and into the I-product

In this section we treat multiary functions *from* the I-product and multivalued functions *into* the I-product.

5.1. Functions from the I-product

There are essentially two views of polyadic functions: one may either conceive of them as *multiary* functions taking *several* parameters or as *unary* functions taking a *tuple* of arguments. These two conceptions lead to isomorphic domains. First we generalize the notion of strictness from unary to multiary functions.

Definition 5.1. Let $(M_j, \sqsubseteq_j)$ be posets with least elements $\bot_j$ $(j \in [1, n+1])$ and $I \subseteq [1, n]$ $(n \geqslant 1)$. A multiary function $f: M_1 \times \cdots \times M_n \to M_{n+1}$ is called *I-strict* if $f(x_1, \ldots, x_n) = \bot_{n+1}$ whenever $x_i = \bot_i$ for some $i \in I$.

Note that an I-strict function is strict *at least* in the arguments from the index set I; it may, however, also be strict in other arguments.

Example 5.2. Consider the following extensions

$$or, lor, ror, por: \mathscr{B}^{\bot} \times \mathscr{B}^{\bot} \to \mathscr{B}^{\bot}$$

of the dyadic disjunction from the set $\mathscr{B} = \{\mathrm{T}, \mathrm{F}\}$ of Boolean values to the flat poset $\mathscr{B}^{\bot} = \{\bot, \mathrm{T}, \mathrm{F}\}$:

or	⊥	T	F
⊥	⊥	⊥	⊥
T	⊥	T	T
F	⊥	T	F

lor	⊥	T	F
⊥	⊥	⊥	⊥
T	T	T	T
F	⊥	T	F

ror	⊥	T	F
⊥	⊥	T	⊥
T	⊥	T	T
F	⊥	T	F

por	⊥	T	F
⊥	⊥	T	⊥
T	T	T	T
F	⊥	T	F

or is strict in both, *lor* only in the first, and *ror* only in the second argument. In [25, p. 774] these cases are called "bistrict", "left-strict", and "right-strict", respectively. The function *por* is strict in no single argument.

There is a natural isomorphism between a subcpo of I-strict multiary functions and strict unary functions on the I-product.

Proposition 5.3. *Let $(M_j, \sqsubseteq_j, \bot_j)$ $(j \in [1, n+1])$ be complete partial orders and $I \subseteq [1, n]$ an index set. Then we have*

$$\{f: M_1 \times \cdots \times M_n \xrightarrow{I\text{-strict}} M_{n+1} \mid f(\bot_1, \ldots, \bot_n) = \bot_{n+1}\}$$

$$\approx [M_1 \star \overset{I}{\ldots} \star M_n \xrightarrow{\text{strict}} M_{n+1}].$$

Proof. Let $\mathscr{F} = \{f : M_1 \times \cdots \times M_n \xrightarrow{I\text{-strict}} M_{n+1} \mid f(\perp_1, \ldots, \perp_n) = \perp_{n+1}\}$ and $\mathscr{G} = [M_1 \star \overset{I}{\ldots} \star M_n \xrightarrow{\text{strict}} M_{n+1}]$. Define $F : \mathscr{F} \to \mathscr{G}$ by $F(f)(\vec{x}) = f(i^{I0}(\vec{x}))$. Then $F(f)$ is strict for all $f \in \mathscr{F}$, hence F is well-defined. Moreover, define $G : \mathscr{G} \to \mathscr{F}$ by $G(g)(\vec{y}) = g(cons^{0I}(\vec{y}))$. If $\vec{y} = \vec{\perp}$ or $y_i = \perp_i$ for some $i \in I$, then $G(g)(\vec{y}) = \perp_{n+1}$; thus G is well-defined. F and G are continuous with $F \circ G = id_{\mathscr{G}}$ and $G \circ F = id_{\mathscr{F}}$; therefore $\mathscr{F}$ and $\mathscr{G}$ are isomorphic. □

Similarly, the corresponding subcpos of monotone and of continuous functions are isomorphic. For nonempty index sets, the previous proposition can be simplified.

Corollary 5.4. *Let* $(M_j, \sqsubseteq_j, \perp_j)$ $(j \in [1, n+1])$ *be complete partial orders and* $\emptyset \neq I \subseteq [1, n]$ *an index set. Then we have*

$$[M_1 \times \cdots \times M_n \xrightarrow{I\text{-strict}} M_{n+1}] \approx [M_1 \star \overset{I}{\ldots} \star M_n \xrightarrow{\text{strict}} M_{n+1}].$$

In particular, a $[1, n]$-strict multiary function corresponds to a strict unary function on the smash product. Moreover, for $I \neq \emptyset$ every I-strict function $f : M_1 \times \cdots \times M_n \to M_{n+1}$ can be uniquely factorized as $f = g \circ cons^{0I}$ with $g : M_1 \star \overset{I}{\ldots} \star M_n \xrightarrow{\text{strict}} M_{n+1}$.

Example 5.5. The extensions of the Boolean disjunction from Example 5.2 may be equivalently regarded as strict unary functions on the corresponding I-products (cf. Example 2.2),

$$or : \mathscr{B}^{\perp} \overset{\{1,2\}}{\star} \mathscr{B}^{\perp} \xrightarrow{\text{strict}} \mathscr{B}^{\perp},$$

$$lor : \mathscr{B}^{\perp} \overset{\{1\}}{\star} \mathscr{B}^{\perp} \xrightarrow{\text{strict}} \mathscr{B}^{\perp},$$

$$ror : \mathscr{B}^{\perp} \overset{\{2\}}{\star} \mathscr{B}^{\perp} \xrightarrow{\text{strict}} \mathscr{B}^{\perp},$$

$$por : \mathscr{B}^{\perp} \overset{\emptyset}{\star} \mathscr{B}^{\perp} \xrightarrow{\text{strict}} \mathscr{B}^{\perp},$$

with the following tables:

or	
$\langle \perp, \perp \rangle$	$\perp$
$\langle T, T \rangle$	T
$\langle T, F \rangle$	T
$\langle F, T \rangle$	T
$\langle F, F \rangle$	F

lor	
$\langle \perp, \perp \rangle$	$\perp$
$\langle T, \perp \rangle$	T
$\langle F, \perp \rangle$	$\perp$
$\langle T, T \rangle$	T
$\langle T, F \rangle$	T
$\langle F, T \rangle$	T
$\langle F, F \rangle$	F

ror	
$\langle \perp, \perp \rangle$	$\perp$
$\langle \perp, T \rangle$	T
$\langle \perp, F \rangle$	$\perp$
$\langle T, T \rangle$	T
$\langle T, F \rangle$	T
$\langle F, T \rangle$	T
$\langle F, F \rangle$	F

por	
$\langle \perp, \perp \rangle$	$\perp$
$\langle \perp, T \rangle$	T
$\langle \perp, F \rangle$	$\perp$
$\langle T, \perp \rangle$	T
$\langle F, \perp \rangle$	$\perp$
$\langle T, T \rangle$	T
$\langle T, F \rangle$	T
$\langle F, T \rangle$	T
$\langle F, F \rangle$	F

Note that *por* viewed as a unary function is strict, whereas viewed as a dyadic function, *por* is strict in no single argument.

Monotonicity and continuity of multiary functions *from* the I-product can be checked for each argument separately.

Definition 5.6. Let $(M_j, \sqsubseteq_j)$ $(j \in [1, n+1])$ be posets with least elements $\perp_j$, $I \subseteq [1, n]$, and $f: M_1 \star \overset{I}{\ldots} \star M_n \to M_{n+1}$ a function $(n \geqslant 2)$. For all $j \in [1, n]$ and $x_1 \in M_1, \ldots, x_{j-1} \in M_{j-1}, x_{j+1} \in M_{j+1}, \ldots, x_n \in M_n$ the *jth argument function*

$$f_{x_1 \cdots x_{j-1} x_{j+1} \cdots x_n}: M_j \to M_{n+1}$$

of f is given by $f_{x_1 \cdots x_{j-1} x_{j+1} \cdots x_n}(x_j) = f(cons^I(x_1, \ldots, x_n))$.

Proposition 5.7. *Let $(M_j, \sqsubseteq_j, \perp_j)$ $(j \in [1, n+1])$ be complete partial orders, $I \subseteq [1, n]$, and $f: M_1 \star \overset{I}{\ldots} \star M_n \to M_{n+1}$ a function $(n \geqslant 2)$. Then f is monotone (continuous) iff all argument functions of f are monotone (continuous).*

Proof. Straightforward. □

5.2. Functions into the I-product

For multivalued functions leading *into* the I-product, the situation is comparably simple: The projections factorize the result tuple, and everything can be checked componentwise.

Proposition 5.8. *Let $(M_j, \sqsubseteq_j, \perp_j)$ $(j \in [0, n])$ be complete partial orders, $I \subseteq [1, n]$, and $f: M_0 \to M_1 \star \overset{I}{\ldots} \star M_n$ a function.*

(a) *If f is strict, then $\pi_j \circ f$ is strict for all $j \in [1, n]$.*

(b) *If $\pi_i \circ f$ is strict for some $i \in I$, then f is strict.*

(c) *f is monotone (continuous) iff all $\pi_j \circ f$ $(j \in [1, n])$ are monotone (continuous).*

Proof. Straightforward. □

6. *I*-strict continuous extensions of operations on sets

The basic operations like $+$, $-$, $\times$, $\wedge$, $\vee$ of the data structures underlying a programming language are usually defined as multiary functions between sets. For the denotational description, these carrier sets are enlarged to flat posets by adjoining a $\perp$-element, and the operations are extended into monotone and hence continuous functions.

With unary operations, there is the choice only for constant functions whether to extend them in a strict or nonstrict way. However, multiary operations in general possess a variety of possible nonstrict extensions. Besides in the naturally (=strictly) extended functions one often is also interested in more defined nonstrict extensions, compare the extensions *lor*, *ror*, and *por* of the disjunction in Example 5.2. Thus, we now investigate how an n-ary operation

$$g: M_1 \times \cdots \times M_n \to M_{n+1}$$

on sets M_j can be extended into an I-strict monotone and hence continuous function on the cartesian product of the flat posets $M_j^{\perp_j} = M_j \cup \{\perp_j\}$ $(\perp_j \notin M_j)$ $(j \in [1,n])$.

Definition 6.1. A function $f: M_1^{\perp_1} \times \cdots \times M_n^{\perp_n} \to M_{n+1}^{\perp_{n+1}}$ is called an *extension* of an operation $g: M_1 \times \cdots \times M_n \to M_{n+1}$ (denoted by $g = f \mid M_1 \times \cdots \times M_n$), if for all $\vec{x} \in M_1 \times \cdots \times M_n$ one has $f(\vec{x}) = g(\vec{x})$. The set of all I-strict continuous extensions of an operation g is denoted by $\mathscr{E}^I(g)$.

Of course, $\mathscr{E}^1(g)$ is partially ordered as a subset of the respective function space.

Corollary 6.2. *For every operation $g: M_1 \times \cdots \times M_n \to M_{n+1}$ and index set $I \subseteq [1,n]$, the set $\mathscr{E}^I(g)$ forms a subcpo of the function space $[M_1^{\perp_1} \times \cdots \times M_n^{\perp_n} \xrightarrow{\text{continuous}} M_{n+1}^{\perp_{n+1}}]$.*

Proof. The natural extension $\hat{g}: M_1^{\perp_1} \times \cdots \times M_n^{\perp_n} \to M_{n+1}^{\perp_{n+1}}$ given by

$$\hat{g}(\vec{x}) = \begin{cases} \perp_{n+1} & \text{if } x_j = \perp_j \text{ for some } j \in [1,n], \\ g(\vec{x}) & \text{if } x_j \in M_j \text{ for all } j \in [1,n]. \end{cases}$$

is the least element in $\mathscr{E}^I(g)$. Moreover, $\mathscr{E}^I(g)$ is closed under forming suprema: Let $G \subseteq \mathscr{E}^I(g)$ be directed and $\vec{x} \in M_1^{\perp_1} \times \cdots \times M_n^{\perp_n}$ with $x_i = \perp_i$ for some $i \in I$. Then $(\bigsqcup G)(\vec{x}) = \bigsqcup_{n+1} G(\vec{x}) = \perp_{n+1}$; hence, $\bigsqcup G$ is I-strict as well. □

The cpos used in the denotational description of programming languages are upward-complete only with respect to directed sets. Hence, for arbitrary subsets, least upper bounds in general do not exist. Nevertheless, the subcpo $\mathscr{E}^I(g)$ possesses both a join and a meet operation.

Proposition 6.3. *Let M_j $(j \in [1, n+1])$ be nonempty sets, $g: M_1 \times \cdots \times M_n \to M_{n+1}$ an operation and $I \subseteq [1,n]$. The set $\mathscr{E}^I(g)$ forms a distributive lattice under the operations*

$$f \bigsqcup h\colon \vec{x} \mapsto f(\vec{x}) \bigsqcup_{n+1} h(\vec{x}), \qquad f \bigsqcap h\colon \vec{x} \mapsto f(\vec{x}) \bigsqcap_{n+1} h(\vec{x}), \qquad (f, h \in \mathscr{E}^I(g)).$$

Proof. Let $T = M_1 \times \cdots \times M_n$ be the set of total and $P = M_1^{\perp_1} \times \cdots \times M_n^{\perp_n} \setminus T$ the set of partial tuples. For $\vec{x} \in M_1^{\perp_1} \times \cdots \times M_n^{\perp_n}$ let $total(\vec{x}) = \{\vec{y} \in T \mid \vec{x} \sqsubseteq \vec{y}\}$.

(a) The function $f \sqcup h$ is well defined: If $\vec{x} \in T$, then $f(\vec{x}) = g(\vec{x}) = h(\vec{x})$; hence $f(\vec{x}) \sqcup_{n+1} h(\vec{x})$ exists. If $\vec{x} \in P$ and $x_i = \perp_i$ for some $i \in I$, then $f(\vec{x}) = \perp_{n+1} = h(\vec{x})$, hence $f(\vec{x}) \sqcup_{n+1} h(\vec{x})$ exists. Otherwise, $\vec{x} \in P$ and $x_i \neq \perp_i$ for all $i \in I$. For every $\vec{y} \in total(\vec{x})$, we have $f(\vec{x}) \sqsubseteq_{n+1} f(\vec{y})$ and $h(\vec{x}) \sqsubseteq_{n+1} h(\vec{y})$ by monotonicity. Thus $f(\vec{x}), h(\vec{x}) \sqsubseteq_{n+1} g(\vec{y})$; hence $\{f(\vec{x}), h(\vec{x})\}$ forms a chain in the flat poset $M_{n+1}^{\perp_{n+1}}$, and $f(\vec{x}) \sqcup_{n+1} h(\vec{x})$ exists.

(b) For all $f, h \in \mathscr{E}^I(g)$ we have $f \sqcup h \in \mathscr{E}^I(g)$: Obviously $f \sqcup h$ is an I-strict extension of g. Moreover, $f \sqcup h$ is monotone, since, for all $\vec{x}, \vec{y} \in M_1^{\perp_1} \times \cdots \times M_n^{\perp_n}$ with $\vec{x} \sqsubseteq \vec{y}$ we have $f(\vec{x}) \sqcup_{n+1} h(\vec{x}) \sqsubseteq_{n+1} f(\vec{y}) \sqcup_{n+1} h(\vec{y})$.

(c) The function $f \sqcap h$ is well defined, since $M_{n+1}^{\perp_{n+1}}$ is flat.

(d) For all $f, h \in \mathscr{E}^I(g)$ we have $f \sqcap h \in \mathscr{E}^I(g)$: Obviously, $f \sqcap h$ is an I-strict extension of g. Moreover, $f \sqcap h$ is monotone, since, for all $\vec{x}, \vec{y} \in M_1^{\perp_1} \times \cdots \times M_n^{\perp_n}$ with $\vec{x} \sqsubseteq \vec{y}$ we have $f(\vec{x}) \sqcap_{n+1} h(\vec{x}) \sqsubseteq_{n+1} f(\vec{y}) \sqcap_{n+1} h(\vec{y})$.

(e) Let $k, l, m \in \mathscr{E}^I(g)$ and $\vec{x} \in M_1^{\perp_1} \times \cdots \times M_n^{\perp_n}$. In the flat poset $M_{n+1}^{\perp_{n+1}}$ $k(\vec{x}) \sqcap_{n+1} (l(\vec{x}) \sqcup_{n+1} m(\vec{x})) = (k(\vec{x}) \sqcap_{n+1} l(\vec{x})) \sqcup_{n+1} (k(\vec{x}) \sqcap_{n+1} m(\vec{x}))$ holds, since by (a) and (c) the respective lubs and glbs exist. This shows $k \sqcap (l \sqcup m) = (k \sqcap l) \sqcup (k \sqcap m)$. □

The lattice $\mathscr{E}^I(g)$ has a greatest element.

Theorem 6.4. *Let $g: M_1 \times \cdots \times M_n \to M_{n+1}$ be an operation on nonempty sets M_j and $I \subseteq [1, n]$. Then g possesses a greatest continuous I-strict extension*

$$\bar{g}^I : M_1^{\perp_1} \times \cdots \times M_n^{\perp_n} \to M_{n+1}^{\perp_{n+1}}.$$

Proof. Let again $T = M_1 \times \cdots \times M_n$ be the set of total and $P = M_1^{\perp_1} \times \cdots \times M_n^{\perp_n} \setminus T$ the set of partial tuples. Put $Q = \{\vec{x} \in P \mid x_i = \perp_i \text{ for some } i \in I\}$; for $\vec{x} \in M_1^{\perp_1} \times \cdots \times M_n^{\perp_n}$ let $total(\vec{x}) = \{\vec{y} \in T \mid \vec{x} \sqsubseteq \vec{y}\}$. Define $\bar{g}^I : M_1^{\perp_1} \times \cdots \times M_n^{\perp_n} \to M_{n+1}^{\perp_{n+1}}$ by

$$\bar{g}(\vec{x}) = \begin{cases} g(\vec{x}) & \text{if } \vec{x} \in T, \\ c & \text{if } \vec{x} \in P \setminus Q \text{ and } g(total(\vec{x})) = \{c\}, \\ \perp_{n+1} & (\text{if } \vec{x} \in P \setminus Q \text{ and } |g(total(\vec{x}))| \geqslant 2) \quad \text{or } \vec{x} \in Q. \end{cases}$$

$\bar{g}^I$ is an *extension* of g, since $\bar{g}^I | T = g$.

$\bar{g}^I$ is *monotone*: Let $\vec{x}, \vec{y} \in M_1^{\perp_1} \times \cdots \times M_n^{\perp_n}$ with $\vec{x} \sqsubseteq \vec{y}$. If $\vec{x} \in T$, then $\vec{x} = \vec{y}$ and $\bar{g}^I(\vec{x}) = \bar{g}^I(\vec{y})$. If $\vec{x} \in P \setminus Q$ and $g(total(\vec{x})) = \{c\}$, then $total(\vec{y}) \subseteq total(\vec{x})$ and $\bar{g}^I(\vec{x}) = c = \bar{g}^I(\vec{y})$. Otherwise, $\bar{g}^I(\vec{x}) = \perp_{n+1} \sqsubseteq_{n+1} \bar{g}^I(\vec{y})$.

$\bar{g}^I$ is *greatest*: Let h be another monotone I-strict extension of g and $\vec{x} \in M_1^{\perp_1} \times \cdots \times M_n^{\perp_n}$. If $\vec{x} \in T$, then $h(\vec{x}) = g(\vec{x}) = \bar{g}^I(\vec{x})$. If $\vec{x} \in P \setminus Q$ and $g(total(\vec{x})) = \{c\}$, then for $\vec{y} \in total(\vec{x})$ we have $h(\vec{x}) \sqsubseteq_{n+1} h(\vec{y}) = c = \bar{g}^I(\vec{x})$ by monotonicity of h. If $\vec{x} \in P \setminus Q$ and $|g(total(\vec{x}))| \geqslant 2$, then there exist $\vec{y}, \vec{z} \in total(\vec{x})$ with $g(\vec{y}) \neq g(\vec{z})$. By monotonicity it follows $h(\vec{x}) \sqsubseteq_{n+1} h(\vec{y}) = g(\vec{y})$ and $h(\vec{x}) \sqsubseteq_{n+1} h(\vec{z}) = g(\vec{z})$. Since $M_{n+1}^{\perp_{n+1}}$ is flat, we have $h(\vec{x}) = \perp_{n+1} \sqsubseteq_{n+1} \bar{g}^I(\vec{x})$. Otherwise, $\vec{x} \in Q$, then $h(\vec{x}) = \perp_{n+1} = \bar{g}^I(\vec{x})$. □

Corollary 6.5. $(\mathscr{E}^I(g), \sqcap, \sqcup)$ *forms a complete lattice.*

Proof. For every nonempty subset $G \subseteq \mathscr{E}^I(g)$, the greatest lower bound $\bigsqcap G$: $\vec{x} \mapsto \bigsqcap_{n+1} G(\vec{x})$ exists in $\mathscr{E}^I(g)$. By Theorem 6.4, $\mathscr{E}^I(g)$ has a greatest element; hence the lattice $(\mathscr{E}^I(g), \bigsqcap, \bigsqcup)$ is complete. □

Examples 6.6. (a) For the constant function *true*: $\mathscr{B} \times \mathscr{B} \to \mathscr{B}$ on Boolean values with $true(x, y) = \mathrm{T}$ for all $x, y \in \mathscr{B}$, the lattice $\mathscr{E}^{\emptyset}(true)$ has $1 + \sum_{i=0}^{4} \binom{4}{i} = 17$ elements.

(b) For the disjunction $\vee : \mathscr{B} \times \mathscr{B} \to \mathscr{B}$ the lattice $\mathscr{E}^{\emptyset}(\vee)$ is isomorphic to the power set $\mathscr{P}(\{1, 2\})$. Its $2^2 = 4$ elements *or*, *lor*, *ror*, *por* were introduced in Example 5.2.

(c) For the addition $+ : \mathscr{N} \times \mathscr{N} \to \mathscr{N}$ of natural numbers, the lattice $\mathscr{E}^I(+)$ degenerates for every $I \subseteq [1, 2]$ to the singleton set $\{\hat{+}\}$ comprising just the strictly extended addition function.

(d) For the multiplication $\times : \mathscr{N} \times \mathscr{N} \to \mathscr{N}$ of natural numbers, the lattice $\mathscr{E}^{\emptyset}(\times)$ is isomorphic to the power set $\mathscr{P}(\{1, 2\})$. The four extensions only differ in the values of the products $0 \times \bot$ and $\bot \times 0$.

The I-strict extensions belonging to different index sets can easily be related.

Proposition 6.7. *For all operations* $g : M_1 \times \cdots \times M_n \to M_{n+1}$ *and index sets* $I \subseteq J \subseteq [1, n]$, $\mathscr{E}^J(g)$ *is a complete sublattice of* $\mathscr{E}^I(g)$ *and* $\bar{g}^J \sqsubseteq \bar{g}^I$ *holds.*

Proof. From $I \subseteq J$ it follows $\mathscr{E}^J(g) \subseteq \mathscr{E}^I(g)$. For every $F \subseteq \mathscr{E}^J(g)$ we have $(\bigsqcup_{\mathscr{E}^I(g)} F) \in \mathscr{E}^J(g)$ and $(\bigsqcap_{\mathscr{E}^I(g)} F) \in \mathscr{E}^J(g)$. Hence $\mathscr{E}^J(g)$ is a complete sublattice of $\mathscr{E}^I(g)$. Thus, for the unit elements, $\bar{g}^J = \bigsqcup \mathscr{E}^J(g) \sqsubseteq \bigsqcup \mathscr{E}^I(g) = \bar{g}^I$ holds. □

In particular, for $I = \emptyset$ we obtain from Theorem 6.4:

Corollary 6.8. *Every operation* $g : M_1 \times \cdots \times M_n \to M_{n+1}$ *on sets* M_j *possesses a greatest continuous extension* $\bar{g} : M_1^{\perp} \times \cdots \times M_n^{\perp} \to M_{n+1}^{\perp}$.

Example 6.9. Consider the conditional projection $c : \mathscr{B} \times \mathscr{N} \times \mathscr{N} \to \mathscr{N}$ given by

$$c(b, m, n) = \begin{cases} m & \text{if } b = \mathrm{T}, \\ n & \text{if } b = \mathrm{F} \end{cases} \quad (m, n \in \mathscr{N}).$$

The greatest continuous $\{1\}$-strict extension $\bar{c}^{\{1\}}$ is the sequential conditional *cond*: $\mathscr{B}^{\perp_{\mathscr{B}}} \times \mathscr{N}^{\perp_{\mathscr{N}}} \times \mathscr{N}^{\perp_{\mathscr{N}}} \to \mathscr{N}^{\perp_{\mathscr{N}}}$ with

$$cond(b, x, y) = \begin{cases} x & \text{if } b = \mathrm{T}, \\ y & \text{if } b = \mathrm{F}, \\ \bot_{\mathscr{N}} & \text{if } b = \bot_{\mathscr{B}} \end{cases} \quad (x, y \in \mathscr{N}^{\perp_{\mathscr{N}}}).$$

The greatest continuous extension $\bar{c}$ is the parallel conditional *parcond*: $\mathscr{B}^{\perp_{\mathscr{B}}} \times \mathscr{N}^{\perp_{\mathscr{N}}} \times \mathscr{N}^{\perp_{\mathscr{N}}} \to \mathscr{N}^{\perp_{\mathscr{N}}}$ given by

$$parcond(b, x, y) = \begin{cases} x & \text{if } b = \mathrm{T} \quad \text{or} \quad x = y, \\ y & \text{if } b = \mathrm{F} \quad \text{or} \quad x = y, \\ \bot_{\mathscr{N}} & \text{if } b = \bot_{\mathscr{B}} \quad \text{and} \quad x \neq y \end{cases} \quad (x, y \in \mathscr{N}^{\perp_{\mathscr{N}}}).$$

Obviously, *cond* $\sqsubseteq$ *parcond*, since $\{1\} \supseteq \emptyset$. Note that the lattice $\mathscr{E}^{\{1\}}(c)$ is infinite: for every $k \in \mathscr{N}$ the function $c_k : \mathscr{B}^{\perp_{\mathscr{B}}} \times \mathscr{N}^{\perp_{\mathscr{N}}} \times \mathscr{N}^{\perp_{\mathscr{N}}} \to \mathscr{N}^{\perp_{\mathscr{N}}}$ given by

$$c_k(b,x,y) = \begin{cases} x & \text{if } b = \mathrm{T} \quad \text{and} \quad (x = k \text{ or } y \neq \perp_{\mathscr{N}}), \\ y & \text{if } b = \mathrm{F} \quad \text{and} \quad (y = k \text{ or } x \neq \perp_{\mathscr{N}}), \\ \perp_{\mathscr{N}} & \text{if } b = \perp_{\mathscr{B}} \text{ or } \quad (x \neq k \text{ and } y = \perp_{\mathscr{N}}) \\ & \qquad\quad \text{or } \quad (y \neq k \text{ and } x = \perp_{\mathscr{N}}) \end{cases}$$

is an element of $\mathscr{E}^{\{1\}}(c)$.

Observe that *partial* operations on sets in general do not possess a greatest I-strict continuous extension on the corresponding flat posets since the different extensions may be incomparable.

Example 6.10. The nowhere defined partial function $\Omega : \mathscr{B} \times \mathscr{B} \rightharpoonup \mathscr{B}$ has exactly 16 maximal continuous extensions to $\mathscr{B}^{\perp} \times \mathscr{B}^{\perp} \to \mathscr{B}^{\perp}$ which are pairwise incomparable.

7. Applications

Finally, we outline some applications of the I-product to the denotational description of programming languages and to the solution of recursive domain equations.

7.1. Call-by-value versus call-by-name

As our first application, we revisit the well-known difference between call-by-value and call-by-name. Consider a (possibly recursive) routine declaration (for the notation see [2])

$$\texttt{funct f} \equiv (\mathtt{s}_1\,\mathtt{x}_1, \ldots, \mathtt{s}_n\,\mathtt{x}_n)\mathtt{s}_{n+1} : \mathtt{e}$$

where the $\mathtt{s}_j$ are sorts like `bool`, `nat` or `string` $(1 \leqslant j \leqslant n+1)$, the $\mathtt{x}_k$ are pairwise different parameters $(1 \leqslant k \leqslant n)$, and the expression `e` of sort $\mathtt{s}_{n+1}$ constitutes the body of the abstraction. In a call-by-value semantics, upon a call $\mathtt{f}(\mathtt{e}_1, \ldots, \mathtt{e}_n)$ all argument expressions $\mathtt{e}_k$ are evaluated first; then their values are passed to the body `e` of the abstraction. If the computation of some argument expression $\mathtt{e}_k$ diverges, then the computation of the entire function application $\mathtt{f}(\mathtt{e}_1, \ldots, \mathtt{e}_n)$ diverges as well. Denotationally, the value of a nonterminating computation is modeled by $\perp$. Thus the interpretation $[\![\,\mathtt{f}\,]\!]$ of `f` is a $[1,n]$-strict function, or equivalently a strict function

$$[\![\,\mathtt{f}\,]\!] : S_1^{\perp_1} \otimes \cdots \otimes S_n^{\perp_n} \xrightarrow{\text{strict}} S_{n+1}^{\perp_{n+1}}$$

on the smash product, where S_j are the carriers associated with the sorts $\mathtt{s}_j$ $(1 \leqslant j \leqslant n)$. In contrast, in a call-by-name semantics, the argument expressions $\mathtt{e}_k$ are substituted

into the body e of the routine; then the body is evaluated. Thus if some argument expression e_k does not occur (decisively) in the body e, the evaluation of the application $f(e_1, \ldots, e_n)$ can terminate although the computation of some e_k diverges. Denotationally, the routine f denotes a nonstrict function

$$[\![f]\!] : S_1^{\perp_1} \times \cdots \times S_n^{\perp_n} \to S_{n+1}^{\perp_{n+1}}$$

on the cartesian product.

Call-by-value and call-by-name can be incorporated into one and the same programming language by providing the following more general function abstraction: For each parameter x_k in an abstraction, it is specified syntactically whether the argument expression itself or its value should be passed in an application; for example as

$$\text{funct } f \equiv (s_1\, x_1, \text{value } s_2\, x_2, \ldots, s_n\, x_n) s_{n+1} : e.$$

Such a generalized function declaration denotes an I-strict function or equivalently a strict function

$$[\![f]\!] : S_1^{\perp_1} \star \overset{I}{\ldots} \star S_n^{\perp_n} \xrightarrow{\text{strict}} S_{n+1}^{\perp_{n+1}}$$

on the I-product. Here the index set I contains exactly those parameter positions that are transmitted by value. This generalized abstraction allows the programmer to use the parameter mechanisms of routines deliberately in order to exploit the benefits both of call-by-value and of call-by-name.

This idea can be traced back to ALGOL 60 (cf., for example, [17, p. 180]) where formal parameters to be called by value where designated in the procedure heading by the key word value.

7.2. *Recursive domain equations*

As a second application we consider recursive domain equations (see [25]). For example, sequences $\mathscr{S}$ of natural numbers are specified by the equation (cf., for example, [18, Ch. 11])

$$\mathscr{S} = \mathscr{E} \oplus \mathscr{N}^{\perp} \times \mathscr{S},$$

where $\mathscr{E} = \{\perp_{\mathscr{S}}, \langle\rangle\}$ comprises the undefined sequence and the empty sequence, and $\oplus$ denotes the amalgamated direct sum. Using the I-product, the generalized domain equation reads

$$\mathscr{S} = \mathscr{E} \oplus \mathscr{N}^{\perp} \overset{I}{\star} \mathscr{S} \quad (I \subseteq \{1, 2\}).$$

For $I = \{1, 2\}$ we obtain the domain

$$\mathscr{S} = \mathscr{N}^{\star} \cup \{\perp_{\mathscr{S}}\}$$

of finite sequences of proper natural numbers with the flat order

$$s \sqsubseteq t \text{ iff } s = \perp_{\mathscr{S}} \text{ or } s = t \quad (s, t \in \mathscr{S})$$

having no limit points.

For $I = \{1\}$ we obtain the domain

$$\mathscr{S} = \mathscr{N}^{\star} \times \{\perp_{\mathscr{S}}\} \cup \mathscr{N}^{\star} \cup \mathscr{N}^{\infty}$$

of partial, finite, and infinite streams of proper natural numbers with the order

$$s \sqsubseteq t \quad \text{iff} \quad s = \langle \hat{s}, \perp_{\mathscr{S}} \rangle \text{ and}$$
$$(t = \langle \hat{t}, \perp_{\mathscr{S}} \rangle \text{ and } \hat{s} \trianglelefteq \hat{t} \quad \text{or} \quad t \in \mathscr{N}^{\star} \cup \mathscr{N}^{\infty} \quad \text{and } \hat{s} \trianglelefteq t)$$

where $\trianglelefteq$ denotes the prefix relation on $\mathscr{N}^{\star} \cup \mathscr{N}^{\infty}$:

$$x \trianglelefteq y \quad \text{iff} \quad \exists r \in \mathscr{N}^{\star} \cup \mathscr{N}^{\infty}: \quad x \& r = y \quad (x, y \in \mathscr{N}^{\star} \cup \mathscr{N}^{\infty})$$

For $I = \{2\}$ the solution is the nonflat one, but limit-free domain

$$\mathscr{S} = (\mathscr{N}^{\perp})^{\star} \cup \{\perp_{\mathscr{S}}\}$$

of finite tuples of possibly undefined natural numbers equipped with the componentwise order

$$s \sqsubseteq t \quad \text{iff} \quad s = \perp_{\mathscr{S}} \text{ or}$$
$$s, t \in (\mathscr{N}^{\perp})^{\star} \quad \text{and} \quad |s| = |t| \text{ and } \forall j \in [1, |s|]: \; s_j = \perp \text{ or } s_j = t_j.$$

Finally, for $I = \emptyset$ the solution is the domain

$$\mathscr{S} = (\mathscr{N}^{\perp})^{\star} \times \{\perp_{\mathscr{S}}\} \cup (\mathscr{N}^{\perp})^{\star} \cup (\mathscr{N}^{\perp})^{\infty}$$

of partial, finite, and infinite streams of possibly undefined natural numbers ordered by

$$s \sqsubseteq t \quad \text{iff} \quad s = \langle \hat{s}, \perp_{\mathscr{S}} \rangle \quad \text{and} \quad (t = \langle \hat{t}, \perp_{\mathscr{S}} \rangle \quad \text{and} \quad \hat{s} \preceq \hat{t} \text{ or}$$
$$t \in (\mathscr{N}^{\perp})^{\star} \cup (\mathscr{N}^{\perp})^{\infty} \text{ and } \hat{s} \preceq t) \text{ or}$$
$$s, t \in (\mathscr{N}^{\perp})^{\star} \text{ and } |s| = |t| \text{ and } s \preceq t \text{ or}$$
$$s, t \in (\mathscr{N}^{\perp})^{\infty} \text{ and } s \preceq t,$$

where $\preceq$ denotes the less-defined prefix relation on $(\mathscr{N}^{\perp})^{\star} \cup (\mathscr{N}^{\perp})^{\infty}$:

$$x \preceq y \quad \text{iff } |x| \leqslant |y| \quad \text{and} \quad \forall j \in [1, |x|]: \; x_j = \perp \text{ or } x_j = y_j$$

8. Concluding remarks

Originally, the denotational semantics of programming languages was used as a technique for formally assigning meanings to programs [12, 19], for introductions

see [7] or [27]. Beyond this interpretation aspect, denotational semantics in the meantime has grown into a method of language development (cf. [18]). Also implementations aiming at a rapid prototyping of language definitions can be derived from the denotational description. In order to reflect the semantic design decisions of a programming language properly, one needs domain constructions that allow expressing flexibly particular strictness constraints.

A research similar to this paper can also be carried out for the direct sum of domains. Here the coalesced resp. the amalgamated sum corresponds to the direct sum where all the least elements of the components are kept separate resp. are all identified. These two domain constructions are just two borderline cases of a more general situation. Also when forming the direct sum of domains, an arbitrary subset of the least elements of the components may be identified. This shows that the identification of certain subsets of least elements in domain constructions is quite a general issue.

Appendix. Basic definitions and results

A *partial order* $(M, \sqsubseteq)$, for short *poset*, consists of a nonempty set M and a reflexive, transitive, and antisymmetric relation $\sqsubseteq$ on M. It is called *flat* or *discrete*, if there is an element $\perp$ in M such that for all $x, y \in M$, $x \sqsubseteq y$ holds iff $x = \perp$ or $x = y$. The order relation reflects the growth of computational information; the least element $\perp$ corresponds to "no information" and denotes program (part)s with aborting or nonterminating computations.

A function $f: M_1 \to M_2$ between posets is called *monotone*, if it preserves the order, i.e. $f(x) \sqsubseteq_2 f(y)$ holds whenever $x \sqsubseteq_1 y$. For monotone functions the information contents of the result increases as the information contents of the argument grows.

A function $f: M_1 \to M_2$ between posets with least elements $\perp_j$ is called *strict*, if it preserves the least element, i.e. $f(\perp_1) = \perp_2$. Thus, a strict function needs some information about its argument to yield a nontrivial result. Strict functions *from* flat partial orders are monotone, whereas monotone functions *into* flat partial orders are strict or constant.

An element $x \in M$ is called an *upper* (*lower*) *bound* of a subset $S \subseteq M$, denoted by $S \sqsubseteq x$ ($x \sqsubseteq S$), if $s \sqsubseteq x$ ($x \sqsubseteq s$) holds for all $s \in S$. A subset S of a poset M is called *bounded* or *consistent* in M, if it has an upper bound in M. An element of M is called *least upper bound* of a subset S and denoted by $\bigsqcup S$, if it is the least element in the set of upper bounds of S in M. The *greatest lower bound*, denoted by $\bigsqcap S$ if existing, is defined dually. A nonempty subset D of M is called *directed*, if every finite nonempty subset $S \subseteq D$ is bounded in D. $(M, \sqsubseteq, \perp)$ is called a *complete partial order*, for short *cpo*, if $(M, \sqsubseteq)$ is a poset with least element $\perp$ where the least upper bound $\bigsqcup D$ exists for every directed subset $D \subseteq M$. The directedness of sets is preserved under monotone functions. Moreover, posets having only finite directed sets are complete; in particular every flat poset is complete.

For algorithmic language constructs one can concentrate on functions which preserve the least upper bounds of directed sets. Accordingly, a function $f: M_1 \to M_2$ between complete partial orders is called *continuous* if for all directed sets $D \subseteq M_1$ the supremum $\bigsqcup_2 f(D)$ exists and $f(\bigsqcup_1 D) = \bigsqcup_2 f(D)$ holds. An equivalent characterization reads: a function $f: M_1 \to M_2$ is continuous iff f is monotone and $f(\bigsqcup_1 D) \sqsubseteq_2 \bigsqcup_2 f(D)$ holds for all directed sets $D \subseteq M_1$. In particular, monotone mappings from flat posets into cpos are continuous.

Two cpos M_1, M_2 are called *isomorphic* (denoted by $M_1 \approx M_2$), if there is an *isomorphism* $f: M_1 \to M_2$, that is, f is bijective, and both f and f^{-1} are monotone. Isomorphisms between complete partial orders are strict.

Often a subset of a cpo constitutes itself a complete partial order under the induced order relation. Thus a subset $S \subseteq M$ of a cpo M together with an element $\bot_S \in S$ and the induced relation $\sqsubseteq_S = \sqsubseteq_M \cap S \times S$ is called a *subcpo* of M, if $(S, \sqsubseteq_S, \bot_S)$ forms a complete partial order and $\bigsqcup_M D = \bigsqcup_S D$ holds for all directed sets $D \subseteq S$. Equivalently, a subset S of M forms a subcpo iff S has a least element $\bot_S$ and $\bigsqcup_M D \in S$ holds for all directed sets $D \subseteq S$. The monotonicity resp. continuity of functions *into* a subcpo can easily be checked by referring to the entire cpo.

For a set M_1 and a cpo M_2, the *function space* $[M_1 \to M_2]$ forms itself a cpo under the order relation induced by M_2: $f \sqsubseteq g$ holds iff $f(x) \sqsubseteq_2 g(x)$ holds for all $x \in M_1$. The least upper bound of a directed set F of functions is determined pointwise: $\bigsqcup F$ exists iff $\bigsqcup_2 F(x)$ exists for all $x \in M$. Moreover, if $\bigsqcup F$ exists, then we have $(\bigsqcup F)(x) = \bigsqcup_2 F(x)$ for all $x \in M_1$. The subsets $[M_1 \xrightarrow{\text{monotone}} M_2]$ of monotone functions, $[M_1 \xrightarrow{\text{strict}} M_2]$ of strict functions, and $[M_1 \xrightarrow{\text{continuous}} M_2]$ of continuous functions all constitute subcpos of $[M_1 \to M_2]$.

An element x of a cpo M is called *finite* or *compact*, if for all directed sets $D \subseteq M$, $x \sqsubseteq \bigsqcup D$ implies $x \sqsubseteq d$ for some $d \in D$. The set $B(M)$ of finite elements of M is called the *basis* of M. A *limit point* of M is a nonfinite element of M. Thus a finite element represents a finite amount of information. For example, in a flat poset all elements are finite. The finite elements of a complete partial order are also finite w.r.t. any subcpo.

A cpo is called *consistently complete* (also *bounded complete*, see for example [24, p. 110]), if all consistent sets have least upper bounds. For example, every flat poset is consistently complete. The consistent completeness of a cpo is inherited by its subcpos.

For an element x of a cpo M, $B(M, x) = \{y \in B(M) \mid y \sqsubseteq x\}$ denotes the set of *finite approximations* of x in M. M is called *countably algebraic*, if $B(M)$ is countable and $\bigsqcup B(M, x) = x$ holds for all $x \in M$. Thus in a countably algebraic cpo there are only countably many finite elements. Moreover, every limit point is the supremum of its finite approximations. Finally, a *domain* is a consistently complete countably algebraic complete partial order.

A poset $(M, \sqsubseteq)$ forms a *lattice* $(M, \bigsqcup, \bigsqcap)$, if for all $x, y \in M$, $x \bigsqcup y$ and $x \bigsqcap y$ exist. Conversely, let $(M, \smile, \frown)$ be an algebra with two dyadic operations fulfilling the laws of commutativity, associativity, and absorption. Then the associated partial order $(M, \sqsubseteq)$ can be retrieved by setting $x \sqsubseteq y$ iff $x \frown y = x$.

A lattice M is called *complete*, if $\bigsqcup S$ and $\bigsqcap S$ exist for all subsets $S \subseteq M$. Equivalently, M is complete iff M has a greatest element and $\bigsqcap S$ exists for all nonempty subsets $S \subseteq M$. A complete lattice M has a *zero element* $0 = \bigsqcap M$ and a *unit element* $1 = \bigsqcup M$.

A lattice M is called *distributive*, if $x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$ (or equivalently $x \sqcup (y \sqcap z) = (x \sqcup y) \sqcap (x \sqcup z)$) holds for all $x, y, z \in M$. A lattice with a zero element 0 and a unit element 1 is called *complemented* if for all $x \in M$ there is $y \in M$ with $x \sqcap y = 0$ and $x \sqcup y = 1$. Finally, a *Boolean lattice* is a (uniquely) complemented distributive lattice.

Acknowledgment

I would like to thank my colleagues B. Möller (Augsburg) and R. Berghammer (Munich) for valuable comments on the drafts of this paper, and the five anonymous referees for their helpful amendments.

References

[1] J. Backus, Can programming be liberated from the von Neumann style? A functional style and its algebra of programs, *Comm. ACM* **21** (1978) 613–641.

[2] F.L. Bauer, M. Broy, W. Dosch, R. Gnatz, F. Geiselbrechtinger, W. Hesse, B. Krieg-Brückner, A. Laut, T. Matzner, B. Möller, F. Nickl, H. Partsch, P. Pepper, K. Samelson (†), M. Wirsing and H. Wössner, *The Munich Project CIP. Vol. 1: The Wide Spectrum Language CIP-L*, Lecture Notes in Computer Science, Vol. 183 (Springer, Berlin, 1985).

[3] G. Birkhoff, *Lattice Theory*, American Mathematical Society Colloquium Publications XXV, Second Printing (American Mathematical Society, Providence, RI, 3rd ed., 1973).

[4] B.A. Davey and H.A. Priestley, *Introduction to Lattices and Order*, Cambridge Mathematical Textbooks (Cambridge University Press, Cambridge, 1990).

[5] P. Dybjer, Domain algebras, in: J. Paredaens, ed., *Automata, Languages and Programming, 11th Colloq.*, Lecture Notes in Computer Science, Vol. 172 (Springer, Berlin, 1984) 138–150.

[6] J.A. Goguen, J.W. Thatcher, E.G. Wagner and H.J.B. Wright, Initial algebra semantics and continuous algebras, *J. ACM* **24** (1977) 68–95.

[7] M.J.C. Gordon, *The Denotational Description of Programming Languages—an Introduction* (Springer, Berlin, 1979).

[8] C.A. Gunter, Universal profinite domains. *Inform. and Comput.* **72** (1987) 1–30.

[9] C.A. Gunter and D.S. Scott, Semantic Domains, in: J. van Leeuwen, ed., *Handbook of Theoretical Computer Science, Vol. B: Formal Models and Semantics* (Elsevier, Amsterdam/MIT Press, Cambridge, MA, 1990) 633–674.

[10] H. Hermes, *Einführung in die Verbandstheorie*, Zweite Auflage, Die Grundlehren der mathematischen Wissenschaften in Einzeldarstellung, Vol. 73 (Springer, Berlin, 1967).

[11] J. Loeckx and K. Sieber, *The Foundations of Program Verification* (Teubner, Stuttgart/Wiley, Chichester, 2nd ed., 1987).

[12] R.E. Milne and C. Strachey, *A Theory of Programming Language Semantics*, 2 vols (Chapman and Hall, London/Wiley, New York, 1976).

[13] B. Möller and W. Dosch, On the algebraic specification of domains, in: H.-J. Kreowski, ed., *Recent Trends in Data Type Specification, 3rd Workshop on Theory and Applications of Abstract Data Types*, Informatik Fachberichte, Vol. 116 (Springer, Berlin, 1985) 178–195.

[14] P.D. Mosses, Abstract semantic algebras!, in: D. Bjørner, ed., *Formal Description of Programming Concepts II*, Proc. IFIP TC2 Working Conference, Garmisch-Partenkirchen, June 6–8, 1982 (North-Holland, Amsterdam, 1983) 45–70.

[15] P.D. Mosses, Denotational semantics, in: J. van Leeuwen, ed., *Handbook of Theoretical Computer Science, Vol. B: Formal Models and Semantics* (Elsevier, Amsterdam/MIT Press, Cambridge, MA, 1990) 575–631.

[16] G.D. Plotkin, *Algebraic Domains*, Lectures Notes, Department of Computer Science, University of Edinburgh, 1981.

[17] H. Rutishauser, *Description of ALGOL 60*, Handbook for automatic computation, Vol. I, Part a, Die Grundlehren der mathematischen Wissenschaften in Einzeldarstellungen, Vol. 135 (Springer, New York, 1967).

[18] D.A. Schmidt, *Denotational Semantics — A Methodology for Language Development* (Brown, Dubuque, IA, 1988).

[19] D.S. Scott, Outline of a mathematical theory of computation, in: *Proc. 4th Ann. Princeton Conf. on Information Sciences and Systems* (1970) 169–176; Revised and slightly expanded version: Tech. Monograph PRG-2 (Programming Research Group, University of Oxford, 1970).

[20] D.S. Scott, Lectures on a mathematical theory of computation, in: M. Broy and G. Schmidt, eds., *Theoretical Foundations of Programming Methodology*, NATO Advanced Study Institutes Series (Reidel, Dordrecht, 1982) 145–292.

[21] D.S. Scott, Domains for denotational semantics, in: M. Nielson and E.M. Schmidt, eds., *Automata, Languages and Programming, 9th Internat. Colloq.*, Lecture Notes in Computer Science, Vol. 140 (Springer, Berlin, 1982) 577–613.

[22] G. Schmidt, R. Berghammer and H. Zierer, Describing domains with sprouts, *Acta Inform.* **27** (1989) 217–245.

[23] M.B. Smyth, Effectively given domains, *Theoret. Comput. Sci.* **5** (1977) 257–274.

[24] M.B. Smyth, The largest Cartesian closed category of domains, *Theoret. Comput. Sci.* **27** (1983) 109–119.

[25] M.B. Smyth and G.D. Plotkin, The category-theoretic solution of recursive domain equations, *SIAM J. Comput.* **11** (1982) 761–783.

[26] J.E. Stoy, *Denotational Semantics: The Scott–Strachey Approach to Programming Language Theory* (MIT Press, Cambridge, 1977).

[27] R.D. Tennent, The denotational semantics of programming languages, *Comm. ACM* **19** (1976) 437–453.